Benjamin Artes

108432628

The Turing, Enigma, and Human Machines

By 1928 the leader of the *formalist* approach of exploring and defining the foundation of mathematics, David Hilbert, made precise his questions: "…was mathematics *complete…consistent…decidable?*"(Hodges 91) *Complete* in that "every statement could either be proved, or disproved." *Consistent* in that "[an invalid statement] could never be arrived at by a sequence of valid steps." And *decidable* in that there existed a "definite method" that could decide the truth of a given assertion.

By 1931 a "young Austrian mathematician Kurt Gödel…showed incontrovertibly that mathematics as we know it cannot be used to prove itself consistent or complete."(Hodges 92) And in 1936 Alan Turing continued this tradition with "On Computable Numbers, with an Application to the Entscheidungsproblem" which would prove decidability unsolvable and help characterize Hilbert's rigorous formalism as a pipe dream.

And though followed by ~20 years of academic and wartime pursuits, during which Turing helped offset the incredible power of the German's Enigma machine and describe the moral and practical possibilities and complications in the emerging field of computers and artificial intelligence, his machine would stand to be his lasting achievement before his suicide in 1954.

II. Hilbert and the 1930s

Hodges, in pages 80-83, describes the events leading up to and causing Hilbert's attempts around the 1930's. "The distinction between science and mathematics had only been clarified in the late nineteenth century." This freed mathematics from the limitations in the real world, advancing it toward a more "*abstract* point of view" and simultaneously "[creating] something of a crisis within pure mathematics."

In 1899, 10 years after G. Peano standardized Dedekind's arithmetic axioms, Hilbert managed to do what Euclid never could: "[he found] a system of axioms which he could prove would lead to all the theorems of Euclidean geometry, without any appeal to the nature of the physical world." His proof had one crux; it assumed that the theory of 'real numbers' was satisfactory. "From Hilbert's point of view this was not good enough. … As Hibert said, all he had done was 'to reduce everything to the question of consistency for the arithmetical axioms, which is left unanswered.'"

The 1880's had brought with it the work of Gottlob Frege and his *logistic* view of mathematics "in which arithmetic was derived from the logical relationships of the entities in the world, and its consistency guaranteed by a basis in reality."(Hodges 83) Frege defined "each number *n* as the *set* of all collections with *n* members." Using symbolic logic of his own design, a precursor to axiomatic predicate logic, and these sets, Frege was able to prove arithmetical truths. This "contributed significantly toward Frege's goal of constructing an axiomatic theory of

arithmetic."(Leavitt 30-31) However, in 1902 Bertrand Russell contacted Ferge with a problem that "effectively undermined Frege's entire program."

Russell explained that although Frege's sets seemed well behaved for reasonable definitions, it was possible to devise a set that was obviously inconsistent. For this task Russell used an equivalent to the 'set of all sets that don't contain themselves,' which must contain itself because it doesn't and therefore can't contain itself because it does.

"Subsequently Ferge and Russell worked together to try to resolve the paradox…Ferge, However, soon gave up on this ambition."(Leavitt 33) Russell continued work with Alfred North Whitehead and in 1928 they published the three-volume *Principia Mathematica*; this attempt managed to side step, as it was now known, Russell's Paradox, by disallowing talk of 'sets of sets.' In this regard it was a success, though it did not manage to prove the consistency or completeness of arithmetic.

During the course of these events Hilbert's zeal only continued to increase. In 1928, at the International Mathematics Congress, Hilbert again called for the exploration of the foundation of mathematics. His work now was an extension of his work in the late 1890's and his questions were "*about* the [*Principia Mathematica*]." He posed three questions at the 1928 congress: 1) Was mathematics *complete*? Could every statement in math be either proven or disproven? 2) Was it *consistent*? If you followed the rules, could you ever arrive at an incorrect conclusion? 3) Was it

decidable? Was there some magic method that could, given any assertion, decide whether it was true? (Hodges 91)

"On Formally Undecidable Propositions of *Principia Mathematica,*" published in 1931 by the Austrian Kurt Gödel, managed to answer two of these points for Hilbert, although not in the way he had wished. Hilbert had believed the answer to all three questions would be yes. Gödel proved that "mathematics as we know it cannot be used to prove itself *consistent* or *complete*."

Like Turing's paper later, Gödel's methods are just as important as the conclusions that he reaches; they inspired mathematician's later to explore new avenues of attack against a problem as persistently difficult as Hilbert's.

"To begin he posited a system in which arithmetical formulae, theorems, and sequences [of numbers] could be expressed in the form of numbers." He gave each symbol in mathematics('+' , '/' , '=') a number, then he assigned numbers to alphabetic symbols and a number to '0'. Thusly '1+1=2' could be expressed as 's0+s0=ss0' where s stands for 'the successor of', in this case, s0=1. And 's0+s0=ss0' would be converted to the number '7611765776' as demonstrated below:

"Successive prime numbers each raised to the power of" the next number in the converted sequence of numbers, "would now be multiplied together:"

This process was required because it ensured that each mathematical statement was encoded to a *unique* value, or, each value "can be broken down *only* one way." And because of the inclusion of logical operators into his encoding scheme, Godel could also encode meta-mathematical statements. Leavitt gives the example, "2+3=5 is an arithmetical formula."

And finally Gödel used this system to encode a statement that forever broke all hope of a complete and consistent system: "Formula G, for which the Gödel number is g, states that there is a formula with Gödel number g that is not provable within [Principia Mathematica] or any related system." Said another way: This statement is not provable. If it were proven true, it would be false and Hilbert's consistency would be false. If it is left out the system is consistent, but not complete. In his paper, Gödel goes further and proves using this statement that any system cannot use its own mathematics to prove itself consistent or complete.

Both the encoding method and the use of a paradox influenced mathematicians that would come later. But for the time being, there was hope that by using some method outside of normal mathematics, perhaps a method could be found that could prove or disprove anything. Or at least the existence of such a method could be proven.

The non-existence of this method was Turing's goal.

III. Turing's Machine and the Entscheidungsproblem

Turing's paper concerned itself with the decision problem of mathematics. Unfortunately the question as posed by Hilbert was too vague and had to be translated into a question that was more easily provable or disprovable. "Turing…refreshed Hilbert's question by casting it in terms not of proofs, but of computing numbers."(Hodges 154) This dovetailed quite nicely with the central concept in Turing's paper. To answer any questions about the existence of a method such as the decision method, Turing would have to invent a model, not of a computer, but of computing in general. Turing's machine is this model of computing.

Turing's machine consisted of the following parts:

1) Tape: This was the input, output and work space of the computer. An infinitely long one-dimensional roll of tape divided into discrete squares that could be blank or hold some symbol, S_i written by the Tape Head
2) Tape Head: This was the machines view of the tape. It could be at a single box at a time and read or rewrite the symbol contained within the square. The symbol within the square at the Tape Head's current location is called the 'scanned symbol.'
3) State/Configuration register: This was another tape that kept track of the state/configuration the machine was currently in and mapped to the instruction table.

4) Instruction table: This table listed the instructions for a given state as well as the state to enter after the instruction was complete(the next state could be the same as the present state)

The tape for Turing's machine served the same purpose as scratch paper does for a human working on a mathematical problem while simplifying the complexities of motion significantly. Divided into discrete squares, the Tape Head could only move one square left or right. Secondly the Symbols S_i that could be marked on the squares served the same purpose as our mathematical notation. Squares could also be empty and thusly the Tape Head could 'erase'; this is equivalent to S_0 being defined as 'blank' and 'erasure' being defined as writing 'S_0'. His machine could also easily be extended to handle '+' or '*' signs but as Turing himself said: "the subject of this paper is ostensibly computable *numbers*."

…	1		1	0	1	…

Example of Tape

In the course of working on a problem of mathematics many different actions must be taken. For multiplication, carries are often needed, as well as addition at the end, etc. In this case, the current action depends on what the current step of multiplication is. But more generally, the action depends upon which state the machine is presently in. Using states allows for a simplified form of memory that

counteracts the limitation of the Tape Head being at a single square and having a single 'scanned symbol' in memory.

Each state has a set of instructions of the form "Write _; Move ;" followed by the next state. Each states actions can change depending on the 'scanned symbol.' For example, if the scanned symbol is blank(S0) and the state is the 5th state, the action could be write 1(S1); move Left; go to state 6, while for a scanned symbol 1(S1) the action could be write 1(S1); move Right; go to state 5.

The state register keeps track of the current state and after the actions provided by the instruction table are executed it is moved to the next state as indicated by the next state in the instruction table.

…	4	5	6	7	8	…

State Register

State	Action for ' '(S0)	Action for '1'(S1)
5	W:1;L->6	W:1;R->5
6	W:1;L->4	W:0;L->4
…	…	…

Example of state table

Below is an example execution of a Turing machine that adds two numbers represented in unary format:

At this point Turing's machine was capable of computations, but this was not what he set out to build. Much like Gödel's proof, he needed a way to encode the instructions for a machine into something that his machine could examine and reproduce; remember he was trying to build a machine that could solve Hilbert's decision problem. He also needed to build what he called a 'Universal Machine.' A machine that could, given the instructions of another machine, reproduce its output.

And so his next step was to find a way to encode the instructions in the state table into letters, and then into numbers. The first step in such a method is to rewrite the state and instruction table onto a single line with each state/instruction segment separated from the next state/instruction segment by a semi-colon ';'.

Then, each state, i, will be replaced by D followed by i repetitions of A. Each action consists of a write followed by a motion and thus encodings of W for write are not necessary: any inclusion of a symbol S_i implies a write and S_i will be replaced by a D followed by i repetitions of C. A motion R or L continues to work as previously, while N indicates no move. "Accordingly q1S0S1Rq2 would thus be expressed as DADDCRDAA."(Leavitt 79) What is left Turing calls the *standard description*. Example: DADDCRDAA;DAADDRDAAA;DAADCCRDAAAA;…

This *standard description* was then to be turned into a numerical string through the following replacements:

A	C	D	L	R	N	;
1	2	3	4	5	6	7

Example String After Replacement:

313325311731133531117311133225311117311113353 17

After these replacements, Turing calls this string of numbers the *description number* for a given Turing machine. It represents, in a single number, the entire state/instruction set for a given machine. More than this though, *every* Turing machine could be listed in a table. Many entries of this table would be invalid, in Turing's encoding system for example, any entry including an 8 or 9 would be undefined. Also many entries may be machines that do nothing, for example a machine that infinitely moves back and forth between two adjacent squares. Some of these entries will end after a finite time, and some will continue indefinitely. Turing refers to these last two as non-circular and circular respectively.

IV. The Universal Machine

"…could one design a Turing machine that would analyze any *other* Turing machine and decide whether that machine" would run forever or halt sometime after it began? Or, could a Turing machine be designed to decide, given a description number, whether the machine was circular or non-circular? This problem became known as the halting problem and it was just this recasting of Hilbert's original decidability question that Hodge's referred to earlier.

In “On Computable Numbers, with an Application to the Entscheidungsproblem,” Turing fully describes his Universal Machine; it was important to his argument that he did. He includes state/instruction tables for exactly how the machine is to take a description number and execute it’s output onto alternating squares of the tape, leaving the other squares as a place to work. His descriptions of it’s workings are complete, and unfortunately, far more than can be described here. Suffice it to say “In order to operate as a Turing machine, all that the universal machine requires is its description number.” (Leavitt 89)

To attack the halting problem, Turing hypothesizes the existence of a machine D, that can decide whether a description number is circular or not. He then uses a reductio ad absurdum proof to outline its impossibility. “There were other ways of demonstrating [this]… The one he himself favoured[sic] was one which brought out the connection with *self-reference.*”(Hodges) Much as Gödel did in his incompleteness theorem.

D is set to work on our list of all description numbers, of which our Universal Machine and D itself must be included; they are after all Turing Machines and therefore have description numbers. D will print a 1 for circular machines and 0 for circle-free machines.

“Next we draw up a list of the computable sequences generated by the circle-free machines, from first to last.”(Leavitt 91) This list is a list of all numbers that can be computed; a list of the output of all of the non-circular machines. If D can exist then numbers not on this list should not be computable by any ‘definite method.’

Turing at this point uses Cantor's diagonal method to prove this cannot be. From this list of 'all possible computed output' he takes the 1st value of the 1st number, the 2nd value of the 2nd, and so on and so forth.

From this newly computed sequence he adds 1 to each number in the sequence. We now have a new number that is not included in the list of computable numbers, for if it were in the list at position n, it's nth digit would now differ by 1, which is a contradiction. D cannot exist. "After all, the number generated through the diagonal method can be described; then why can it not be computed?"(Leavitt 93)

Turing provides the alternate *self-referential* proof as well:

"Let us image that we can somehow link the deciding machine, D, to the universal machine, U." This machine first runs D on the given description number and if it is circular the DU machine halts. If the machine is circle-free it passes the description number onto U and simulates its output. For this reason it is easy to see that DU is non-circular, by definition.

DU, like any other Turing machine has a description number. If that non-circular description number is given to DU, D decides DU is itself circle-free and

passes its output to U. "It therefore passes the description number onto U, which simulates the action of DU, feeding the description number of DU into D." At this point the simulated D again finds DU to be circle-free and the process repeats, ad infinitum. Therefore DU appears to be both circle-free and circular, "since it is impossible that DU be both, Turing writes 'we conclude that there can be no machine D.'" The halting problem has been solved and with it goes Hilbert's Entscheidungsproblem.

V. Alonzo Church and other solutions

Gödel's proof against completeness and consistency had given the mathematical world new methods and understanding of the form these meta-mathematical proofs were going to take, and in the following 5 years many mathematicians had been attempting Hilbert's final question. And so it is no surprise that Alan Turing was not the first to arrive at the solution to the Entscheidungsproblem.

Alonzo Church, an American logician, showed a paper on lambda-calculus to the American Mathematical Society in 1935 and published it in 1936, just as Alan Turing was finishing drafts of his own paper. Church's approach used lamda-calculus that he had developed and was, "in a certain sense weaker." Lamda-calculus "could be used to translate all the formulae of arithmetic into a standard form. In this form, proving theorems was a matter of converting one string of symbols of the lambda-calculus into another string."(Hodges 112) Church managed to show that in general, lambda-calculus could provide no way of knowing, and no

definite method for converting one string into another, and thus, no absolute knowledge about provability or disprovability. "But it was not obvious that 'a formula of the lambda-calculus' corresponded to the notion of a 'definite method.'"(Hodges 112)

"Turing would have to acknowledge Church before he could publish…and so that August he sketched out, as an appendix to his own paper, a proof of equivalency between his notion of computability and Church's of lamda-definiability."(Leavitt 114) Church's paper was both first and more inline with classical mathematics. Turing's paper was further reaching and more practically convincing, with its images of female computers in a warehouse and mechanical commonsense. Many who read Turing's paper also realized the far-reaching implications of his designing a universal computing machine, the progenitor to our modern computers. Unfortunately for both Church and Turing these results created far smaller waves than did Gödel's. Of mathematician's like Von Neumann, who had spoken to Gödel after his announcement and examined Gödel's work extensively, Leavitt writes "…and when Gödel's findings were published, he abandoned logic altogether in favor of other fields…"(Leavitt 123)

Nonetheless, Turing went to Princeton to work with Church, who was a man with as many personality quirks as Turing himself. Church never took a particular interest in Turing, despite their now sharing the title to the solution of the Entscheidungsproblem – Church-Turing Thesis – "Reminded of Turing by [William] Aspray, Church continued, 'Yes, I forgot about him when I was speaking about my

own graduate students. Truth is, he was not really mine… I did not have enough contact with him to know [much about him].'"

VI. Impact

Written when he was 23, "On Computable Numbers, with an Application to the Entscheidungsproblem" would turn out to be Turing's most well-known accomplishment. During World-War II, he worked at Bletchley Park and was pivotal in managing to consistently crack German messages encoded with the infamous Enigma machine. It was even here that he managed to build his first machines, though they were far less general than the Universal Machine he envisioned.

And yet, this crypto-analytic service to his country was considered top-secret and he never spoke of it in his lifetime. The machines he built were destroyed at the end of the war and he returned for a short time to continue work on his machines.

In 1946 he presented a paper with the first design of a stored program computer and in 1950 he addressed artificial intelligence, going so far as to propose the now named "Turing Test." In this test a person has a conversation with an unknown party through a computer interface. The unknown party may either be another human or an artificial intelligence. If the computer can trick the person into thinking it is human some percentage of times, it can truly be called artificial intelligence. Today this test has been criticized for various problems, but it succeeded in proposing the type of experimental criterion that would be necessary to test for something as hazily defined as intelligence.

VII. Death

In 1952 Alan Turing was convicted of Indecent Behaviour for being homosexual. Possibly because of his wartime efforts he was offered a choice: instead of jail he could instead under go a course of chemical libido adjustment through administered doses of synthetic estrogen.

In 1954, he was found dead in his apartment. Hodges and Leavitt both propose suicide with a cyanide-laced apple, though in recent years other options have been proposed.

VIII. Conclusion

The early 1900's and the 1930's in particular, were a time of incredible expansion for mathematics. It had been divorced from science and new methods and forms of algebra were appearing everywhere. This algebra needed a significantly more solid foundation to rest on and Hilbert's continual push managed to unite mathematicians in this goal. Ferge and Russell began the construction of such systems while at the same time finding the holes and paradoxes that would grow into the questions that Hilbert would succinctly pose to the community at large.

Gödel managed to not only answer the first two of Hilbert's questions, but propose new methods of attacking such meta-mathematical problems at large. These methods directly lead to three solutions to the final problem within five years: Church's lambda-calculus, Emil Post's papers on computing factory's, and most importantly Alan Turing's computing machines.

And though all three managed to answer Hilbert's call, Turing's was the only one that managed to propose, in the course of answering a question on computability in general, that if the methods of computing could finally be divorced from the humans who computed, not only could statements about computing be made, but machines that computed could be constructed as well. In later life Turing and the mathematical community would expand on these ideas, but they were brought to life by the mathematical climate of the 1930's, Hilbert's calls, and Turing's machine-based answer.